МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Построение и анализ алгоритмов»
Тема: Поиск с возвратом

Студент гр. 3381 Сычев Н.С.

Преподаватель Шестопалов Р.П.

Санкт-Петербург
2025

Цель работы.

Цель лабораторной работы заключается в изучении основ метода поиска с возвратом (backtracking), а также в его практическом применении для решения задач, требующих перебора возможных вариантов с возвратом на предыдущие этапы в случае неудачи.

Задание.

У Вовы много квадратных обрезков доски. Их стороны (размер) изменяются от 1 до N−1, и у него есть неограниченное число обрезков любого размера. Но ему очень хочется получить большую столешницу - квадрат размера N. Он может получить ее, собрав из уже имеющихся обрезков(квадратов).

Внутри столешницы не должно быть пустот, обрезки не должны выходить за пределы столешницы и не должны перекрываться. Кроме того, Вова хочет использовать минимально возможное число обрезков.

Входные данные. Размер столешницы - одно целое число $2 \leq N \leq 40$.

Выходные данные. Одно число K, задающее минимальное количество обрезков(квадратов), из которых можно построить столешницу(квадрат) заданного размера N. Далее должны идти K строк, каждая из которых должна содержать три целых числа x, y и w, задающие координаты левого верхнего угла ($1 \leq x, y \leq N$) и длину стороны соответствующего обрезка(квадрата).

Выполнение работы.

В ходе анализа задачи было выяснено, что существует четыре основных варианта для решения задачи в зависимости от размера столешницы N, соответственно:

1. Для столешницы, размер которой кратна 2, квадраты всегда размещаются в определенных координатах, и их размер остается неизменным для всех таких полей. Таких квадратов всегда 4.

2. Если размер столешницы кратен 3, то решение состоит из 6 квадратов.
3. Если размер столешницы кратен 5, то решение состоит из 8 квадратов.
4. В остальных случаях для нахождения оптимального решения используется метод поиска с возвратом (backtracking). Нужно отметить, что на начальном этапе всегда размещается квадрат размером (n / 2+1) и два квадрата (n / 2) по бокам от него. Это помогает оптимизировать алгоритм, сокращая незаполненную площадь столешницы.

Сложность алгоритма.

В случаях, когда размер доски кратен 2, 3 или 5, сложность будет O(1), так как решение вычисляется за константное время.

Но все же алгоритм имеет общую экспоненциальную сложность O(2^N), потому используется рекурсивный поиск с возвратом, обходящий все возможные подмножества множества клеток столешницы.

Основные структуры:

1. Структура Square - структура, которая описывает квадрат, размещающийся на столешнице. Имеет поля *x, y* для координат левого верхнего угла квадрата и поле *w* – размер стороны квадрата.
2. Структура Board - структура, которая представляет собой столешницу и управляет процессом размещения квадратов на ней. Имеет поля: *size* — размер столешницы, *tempX*, *tempY* — временные координаты для поиска свободного места для размещения нового квадрата, *minSquares* — минимальное количество квадратов, необходимых для покрытия поля, *curSquareSize* — текущий размер квадрата, размещаемый на столешнице, *countSquares* — количество квадратов, уже размещённых на столешнице, *board* — двумерный срез, представляющий саму столешницу (где каждая клетка содержит номер квадрата, который ее занимает или 0, если клетка свободна), *bestBoard* — лучшее найденное решение (с минимальным количеством квадратов), *squares* — список

всех размещённых квадратов, *bestSolution* — количество квадратов в лучшем решении.

Описание функций:

1. *void NewBoard(int size)* – функция конструктор для создания поля, принимающая на вход размер поля создаваемого поля. Функция создает пустую матрицу для поля и инициализирует все остальные поля структуры значениями по умолчанию.
2. *void initializeBoard(Board& board)* – функция, принимающая на вход структуру столешницы. Создает внутри нее три начальных квадрата и устанавливает временные координаты *tempX, tempY* на центр столешницы для размещения будущих квадратов.
3. *bool isBoardFull(const Board& board)* – функция принимает на вход структуру столешницы и проверяет, заполнена ли она. Функция пытается найти пустую клетку и начать размещение нового квадрата. Если все возможные квадраты определённого размера не помещаются, функция уменьшает размер квадрата и пытается разместить его заново.
4. *bool newSquare(Board& board, int square_size)* – функция, которая пытается разместить новый квадрат на столешнице с использованием указанного размера квадрата. Она проверяет, можно ли разместить квадрат с текущими размерами, не выходя за пределы поля и не перекрывая уже размещённые квадраты. Если это возможно, квадрат размещается на столешнице и добавляется в список квадратов.
5. *void Backtrace(Board& board)* – функция выполняет откат, если текущее решение не приводит к оптимальному результату. Она удаляет последний размещенный квадрат и уменьшает его размер для повторной попытки.
6. *bool canDeleteSquare(const Board& board, int x, int y)* – функция, проверяющая, можно ли удалить квадрат со столешницы, основываясь на его координатах.

7. *void deleteSquare(Board& board, int x, int y)* – функция удаляет квадрат с заданными координатами со столешницы, если его можно удалить, и уменьшает счётчик квадратов.
8. *void Calculations(Board& board)* – основная функция для поиска оптимального решения. Она инициализирует столешницу и пытается найти минимальное количество квадратов, которое можно разместить. Использует методы *isBoardFull, newSquare* и *Backtrace* для поиска и улучшения решения.
9. *void printCoords(const Board& board)* – функция принимает на вход структуру столешницы и выводит координаты квадратов в лучшем найденном решении.
10. *void process(Board& board, int size)* – функция принимает на вход структуру столешницы и ее размер, распечатывает заранее заданные решения для столешниц размером, кратным 2, 3 или 5. В противном случае вызывает функцию *Calculations* для нахождения оптимального решения
11. Функция *main()* для запуска алгоритма.

Результаты тестирования см. в приложении Б.

Выводы.

В ходе лабораторной работы был изучен метод поиска с возвратом (backtracking), который применялся для решения задачи покрытия столешницы размером N×N наименьшим количеством квадратов с целочисленными размерами сторон. Были рассмотрены оптимальные способы размещения квадратов в зависимости от размера столешницы, а также использован метод поиска с возвратом для нахождения решения.

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Результаты тестирования представлены в табл. Б.1.

Таблица Б.1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	7	9 6 6 2 4 6 2 7 5 1 4 5 1 7 4 1 5 4 2 1 5 3 5 1 3 1 1 4	Время работы: 0.0005 сек. Количество операций: 175
2.	20	4 1 1 10 11 1 10 1 11 10 11 11 10	Время работы: 0.0005 сек. Количество операций: 0
3.	29	14 15 23 7 22 22 8 21 22 1 21 20 2 18 20 3 15 20 3 15 17 3 15 16 1 23 15 7 18 15 5 16 15 2 1 16 14	Время работы: 0.0562 сек. Количество операций: 2626888

		16 1 14 1 1 15	
4.	37	15 25 33 5 26 32 1 25 32 1 19 32 6 30 30 8 27 30 3 19 24 8 19 21 3 19 20 1 27 19 11 22 19 5 20 19 2 1 20 18 20 1 18 1 1 19	Время работы: 0.79651440 сек. Количество операций: 30370889